DOCUMENT RESUME

ED 270 083 IR 012 096

AUTHOR Osborne, Wilma M.
TITLE Executive Guide to Software Maintenance. Reports on Computer Science and Technology.
INSTITUTION National Bureau of Standards (DOC), Washington, D.C.
REPORT NO NBS-SP-500-130
PUB DATE Oct 85
NOTE 33p.; For related documents, see IR 012 097-098.
AVAILABLE FROM Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402.
PUB TYPE Guides - General (050)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Software; Cost Effectiveness; *Maintenance; Microcomputers; Programing; Quality Control
IDENTIFIERS Adaptive Evaluation Structure; *Software Design; Software Evaluation; Software Maintenance; *Software Testing

ABSTRACT

This guide is designed for federal executives and managers who have a responsibility for the planning and management of software projects and for federal staff members who are affected by, or involved in, making software changes, and who need to be aware of steps that can reduce both the difficulty and cost of software maintenance. Organized in a question and answer format, the guide provides answers to 64 questions that address: (1) the feasibility and applicability of software reuse; (2) the development of maintainable software; (3) the improvement of existing software; (4) achieving programmer and software productivity; (5) the three key attributes of maintainable software--correctness, understandability, and reliability; and (6) software configuration management (SCM), including software tools that can aid in making existing code more maintainable. Also included are a list of supporting ICST (Institute for Computer Sciences and Technology) documents and eight suggested additional readings. (JB)

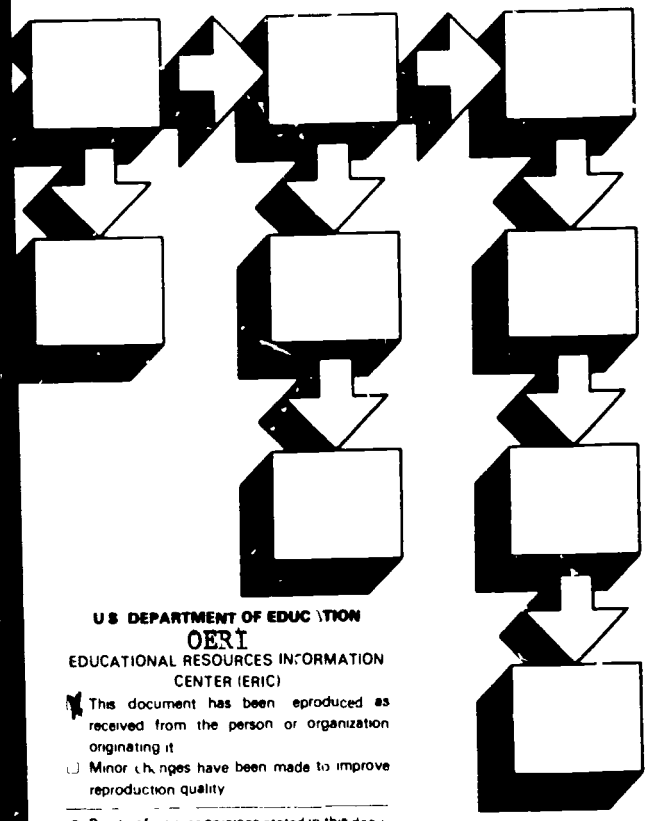
U.S Department of Commerce
National Bureau of Standards

Computer Science and Technology

NBS Special Publication 500-130

Executive Guide to Software Maintenance

Wilma M Osborne



Computer Science and Technology

NBS Special Publication 500-130

Executive Guide to Software Maintenance

Wilma M Osborne

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, MD 20899

Issued October 1985

U.S. DEPARTMENT OF COMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards
Special Publication 500-130

Natl Bur Stand (U.S.), Spec Publ 500-130, 26 pages (Oct 1985)
CODEN XNBSAV

Library of Congress Catalog Card Number:
85-600597

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON 1985

BACKGROUND

This Guide provides answers to sixty-four key questions about software maintenance. It is designed for Federal executives and managers who have a responsibility for the planning and management of software projects. It is also intended for Federal staff members affected by, or involved in, making software changes and who need to be aware of steps that can reduce both the difficulty and cost of software maintenance.

Issues addressed in the Guide include the feasibility and applicability of software reuse, the development of maintainable software, as well as the improvement of existing software, achieving programmer and software productivity, and the three key attributes of maintainable software: correctness, understandability, and reliability. Finally, it discusses software tools that can aid in making existing code more maintainable. The question and answer format is used to organize the material in a concise manner that represents a general approach for evaluating software maintenance problems and alternative workable solutions.

UNDERSTANDING SOFTWARE MAINTENANCE

1. WHAT IS SOFTWARE MAINTENANCE?

Software maintenance is the performance of those activities required to keep a software system operational and responsive after it is accepted and placed into production. It is the set of activities which result in changes to the originally accepted (baseline) product set. These changes consist of modifications created by correcting, inserting, deleting, extending, and enhancing the baseline system.

2. WHAT ARE THE TYPES OF SOFTWAR MAINTENANCE?

The three types of software maintenance are perfectiv adaptive, and corrective.

3. WHAT IS PERFECTIVE MAINTENANCE?

Perfective maintenance includes all changes, insertion deletions, modifications, extensions, and enhancemen which are made to a system to meet the evolvi and/or expanding needs of the user. They are general performed as a result of new or changing requirement or in an attempt to augment or fine tune the softwar Activities designed to make the code easier to unde stand , such as restructuring or documentation updat (often referred to as "preventive" maintenance), a considered to be perfective. Optimization of code make it run faster or use storage more efficiently is al included in the perfective category. Estimates ind cate that perfective maintenance comprises approx mately 60% of all software maintenance efforts.

4. WHAT IS ADAPTIVE MAINTENANCE?

Adaptive maintenance consists of any effort which initiated as a result of changes in the environment which a software system must operate. It accounts f about 20% of all the software maintenance effort The environmental changes are normally beyond th control of the software maintainer and consist primar ly of changes to the:

- o rules, laws, and regulations that affect the system;
- o hardware configurations, e.g., new terminals, local printers;
- o data formats, file structures; and
- o system software, (e.g., operating systems, compilers), utilities.

5. WHAT IS CORRECTIVE MAINTENANCE?

Corrective maintenance refers to changes necessitated by actual errors (induced or residual "bugs") in a system. It accounts for 20% of all the software maintenance efforts and consists of activities normally considered to be error correction required to keep the system operational. The key causes for corrective maintenance are design errors, logic errors, and coding errors. Design errors are generally the result of incorrect, incomplete, or unclear descriptions of the system change being requested, or a misunderstanding of the change request. Logic errors are the result of invalid tests and conclusions, faulty logic flow, incorrect implementation of the design specifications, or unusual combinations of data, which were not thoroughly tested. Coding errors are generally caused by the programmer and are the result of either incorrect implementation of the detailed logic design, or the incorrect use of the source code logic.

6. WHAT IS MAINTAINABILITY?

Maintainability refers to the effort required to find and fix or modify an error in operational software. Maintainability examines the effects of software failure, and ways to minimize those effects. In order to maintain control over the software maintenance process, and to ensure that the maintainability of the system does not deteriorate, it is important that software maintenance be anticipated and planned. The quality and maintainability of a software system often decrease as the system grows older. This is the result of many factors which, taken one at a time, may not seem significant but become cumulative and often result in a system which is very difficult to maintain. Quality programming capabilities and techniques are readily available. However until a firm discipline is placed on how software maintenance is performed, and that discipline

is enforced, many systems will be permitted to deteriorate to the point where they are impossible to maintain. It is likely that as the software maintainability is improved, the capacity to handle additional changes will increase

7. WHAT IS THE "REQUIREMENTS TO RELEASE" CONCEPT?

Generally, changes are made in order to keep the system functioning in an evolving, expanding user and operational environment. When software maintenance is performed as an iterative development process, the concept of requirements to release can be applied. This concept presumes that whenever a modification is made to the software system, a functional review is made to determine if a corresponding change is needed anywhere else in the system This review should, as a minimum, include an assessment of the software design, code, test data, and associated documentation products.

8. HOW CAN "REQUIREMENTS TO RELEASE" HELP TO IMPROVE THE SOFTWARE MAINTENANCE PROCESS?

A significant portion of the activities performed in software maintenance environments is reactive. As a result, there is often a tendency to zero in on the immediate problem, fix it and wait for the next problem to arise. If an examination is made of the functional requirements, there is a likelihood of gaining a better perspective on the problem, and its solution.

9. HOW IS SOFTWARE MAINTENANCE CURRENTLY PERFORMED?

In many organizations, software maintenance is still performed with second generation tools, using second

generation techniques. There are indications, however, that managers are adopting and enforcing the use of improved techniques and practices. Standards and guidelines which provide guidance on how to improve software quality have been published by NBS, IEEE, and the DOD.

10. WHAT IS THE RELATIONSHIP BETWEEN SOFTWARE QUALITY AND SOFTWARE MAINTENANCE?

Software quality can be characterized by such attributes as reliability, understandability, testability, modularity, and expandability, all of which make the software more maintainable. Thus, software quality and maintainability are inextricably bound together. The greater the number of quality attributes engineered into the software during development, the higher the degree of confidence in its maintainability. Although it is significantly easier to build quality into the software, it can also be improved during maintenance. The cost of adding quality to existing software, however, is considerably more expensive.

11. WHAT IS THE RELATIONSHIP BETWEEN SOFTWARE TESTING AND SOFTWARE MAINTENANCE ?

Testing is performed to find errors and omissions. It is also performed to ensure that the logic and structure of the software are appropriate. The more thorough the unit, integration, and system testing, the more likely the software will be reliable and correct. Consequently testing provides assurance that the activities of software maintenance have been performed correctly.

COSTS

12. WHAT IS THE COST OF SOFTWARE MAINTENANCE?

Software maintenance represents 60%-70% of the total cost of software which runs into the tens of billions of dollars each year. Perfective maintenance (changes, enhancements, extensions, etc.) comprises approximately 60% of the software maintenance costs. Adaptive maintenance and corrective maintenance are each approximately 20% of the total.

13. CAN SOFTWARE MAINTENANCE COSTS BE REDUCED?

While total software costs have risen rapidly, the ratio of development to maintenance costs has remained relatively constant. This is a result of the growing dependency of many organizations on software, the extended life of software, and the increasing complexity and size of many applications.

Continued advances in modern programming technology (e.g. modern software engineering tools and methods, fourth generation languages, application generators, etc.) will permit more sophisticated systems to be built at less cost. The dichotomy of this situation is that the more useful a system, the longer it will be used and the more it will cost over its lifetime. Thus, for some software, the cost over the total software lifetime may actually rise, but the cost per year will decrease

14. WHAT CAN BE DONE TO BRING SOFTWARE MAINTENANCE COSTS UNDER CONTROL?

The most important piece in the puzzle of controlling

total software costs is strong, effective management of the entire process The greater the discipline invoked in the software maintenance process, the higher the quality of software which will result

The Federal Government continues to custom develop more than 90% of its software. Reuse of existing software, including the use of off-the-shelf packages, can help hold down costs. Other solutions, such as the use of non-procedural languages; greater use of modern software engineering technology, tools and methods; and the institution of more effective management control also offer opportunities to bring the cost of software maintenance under control.

15. SHOULD AN ORGANIZATION HAVE A SOFTWARE MAINTENANCE POLICY?

Yes, a software maintenance policy is a vital step in controlling software maintenance costs. The policy should describe in broad terms the responsibilities, authorities, functions. and operations of the software maintenance organization.

PERSONNEL

16. WHAT TYPE OF PERSON IS NEEDED FOR SOFTWARE MAINTENANCE?

The person needed in a software maintenance environment should be highly skilled and able to perform all of the functional activities that occur during the software lifecycle. The person should be experienced, and have good analytical capabilities. Above all, these people should be disciplined and thorough in their approach to analysis, coding, debugging, testing, and documentation.

17. WHAT SPECIFIC EXPERIENCES ARI NEEDED FOR SOFTWARE MAINTENANCE

It is essential that software maintainers have experi ence working on tasks individually, as well as in teams The person should have had a major responsibility fo the completion of
some of these tasks It is important that the perso have a breadth of knowledge about software manage ment
since in a maintenance environment, anything tha can go wrong usually does As a minimum, the perso should have a number of years experience working:

- o independently,
- o as part of a team,
- o with specific languages and computers,
- o with modern programming practices,
- o on various types of applications.

18. WHY DO SOFTWARE MAINTAINER HAVF A HIGHER RATE OF TURNOVEI THAN OTHER ADP PERSONNEL?

Maintainers tend to change jobs at a higher rate tha other ADP professionals primarily because of the imag of maintenance. In many organizations, the tradition ally held view that maintenance requires less aptitud than development is quite common. As a result, work ing conditions, including remuneration, tends to be les for maintainers than for developers. Within the nex few years, this situation should change as increasin numbers of managers recognize the value of the main tainer to the daily operation of the organization.

SOFTWARE REUSE

19. WHAT IS REUSABLE SOFTWARE?

The popular notion of software reuse focuses on the software's source and object code. While source and object code can be reused, software consists of more than just code Research has shown that greater benefits can be accrued through the use of the requirements, specifications, design, documentation, test data, and other elements of the software

20. CAN SOFTWARE REUSE IMPROVE SOFTWARE MAINTENANCE?

Yes. In fact, software reuse should be at the heart of the strategy for software maintenance. Research has shown that the quality of software deteriorates when the only elements of the software that are reused are the source and object code. By effectively reusing the requirements, specifications, design, documentation, test data, and other elements on which the code is based, the quality of the software can be maintained, or even enhanced during the repeated modifications which occur after the software has been developed and placed into operational use

21. HOW DO I RECOGNIZE THE OPPORTUNITIES FOR REUSE?

Any time a new application is developed, or enhancements are planned for existing software, the first question which should be addressed is "Do software elements (requirements, specifications, etc.) exist which can be reused in this application?" The type of software reuse that is practical in each situation varies. If a new application is being developed, the opportunity to reuse elements from existing similar application and subsystems should be examined. When enhance-

ments are made to an existing application, existing software elements should be used as the baseline whenever possible. Only when it is clear that reuse of existing software elements is inappropriate, should the decision be made to create brand new software elements.

MANAGEMENT

22. HOW CAN THE IMAGE OF SOFTWARE MAINTENANCE BE IMPROVED?

Upper management must be kept informed of the overall success of the software maintenance effort and how software maintenance supports and enhances the organization's ability to meet its objectives. Software maintenance is an important effort which supports and contributes to the ability of the organization to meet its goals. Too many of the problems encountered in software maintenance are the result of the negative attitude that it is a function which exists because the software support staff can "never do it right." Rather the emphasis should be on the concept that software maintenance enables an organization to improve and expand its capabilities using existing systems.

The software maintenance manager has the responsibility for keeping the maintenance staff happy and satisfied. Software maintenance must be thought of as the challenging, dynamic, interesting work it can be.

23. WHAT ARE SOME OF THE ACTIONS THAT MANAGEMENT CAN TAKE TO DETERMINE WHEN TO CONSIDER REDESIGN?

A system which is in virtually constant need of corrective maintenance is a prime candidate for redesign. As systems age and additional maintenance is performed

on them, many become increasingly fragile and susceptible to changes. The older the code, the more likely frequent modifications, new requirements, and enhancements will cause the system to break down.

When a decision has been reached to redesign or to stop supporting a system, the decision can be implemented in a number of ways: support can simply be removed and the system can die through neglect; the minimum support needed to keep it functioning may be provided while a new system is built; or the system may be rejuvenated section by section and given an extended life. How the redesign is affected depends on the individual circumstances of the system, its operating environment, and the needs of the organization it supports.

24. HOW DO YOU DETERMINE WHEN TO ACQUIRE NEW SOFTWARE?

Although maintenance is an ongoing process, there comes a time when serious consideration should be given to acquiring new software. If there are requirements for which the existing system is totally inadequate or if the existing software is sufficiently outdated that the viability of the organization is affected, then consideration should be given to the acquisition of new software.

A major concern of managers and software engineers is how to determine whether a software package will satisfy all of the existing requirements. In most cases, the software package handles only a part of the requirements. Thus, the acquired software package often will either have to be modified, or additional software will be needed.

25. WHAT ARE THREE KEY QUESTIONS TO BE ANSWERED WHEN ACQUIRING A SOFTWARE PACKAGE?

Organizations are acquiring software packages at an increasing rate. There are a number of questions that should be answered, however, the three that are perhaps the most crucial are:

- o How well can I use the software as it is?
- o How easy is the software to maintain?
- o Will I be able to use it if I change the processing environment?

26. WHAT ARE SOME OTHER FACTORS TO CONSIDER WHEN SELECTING A SOFTWARE PACKAGE?

- o Does the purchase or lease agreement permit the purchaser to make modifications?
- o If not, are there adequate assurances that the seller of the package will make needed modifications?
- o Will the staff require training in order to understand, modify or test the software package?
- o If so, will the training be provided free or at an additional cost to the purchaser?

For further details on the factors to be considered, see [NBS114], [NBS88].

27. HOW DO YOU DECIDE WHETHER TO CONTINUE MAINTENANCE OR TO REDESIGN?

The costs and benefits of the continued maintenance of software which has become error-prone, ineffective and costly must be weighed against those of redesigning the system. While there are no absolute rules on when to rebuild rather than maintain the existing system, some of the conditions that might lead to a deci

sion to redesign include:

- o Frequent failures
- o Code over seven-to-ten years old
- o Overly complex program structure and logic
- o Code written for outdated hardware
- o Running in emulation mode
- o Very large modules or unit subroutines
- o Excessive resource requirements
- o Hard-coded parameters that must be changed
- o Difficulty in keeping maintainers
- o Seriously deficient documentation
- o Missing or incomplete design specifications

The estimated life cycle of a major application system is seven-to-ten years, although 15 to 20 year-old software systems are not uncommon. Software tends to deteriorate with age as a result of numerous fixes and patches. However, if the system was designed and developed in a systematic, maintainable manner, and if maintenance was carefully performed and documented using established standards and guidelines, it may be possible to run it efficiently and effectively for many more years.

28. WHY SHOULD THERE BE A CENTRAL APPROVAL POINT FOR SOFTWARE CHANGES?

A central approval point is essential to ensure that the desired changes are made to the appropriate software. One of the key problems in many software maintenance environments is inadequate control over the change process.

29. SHOULD DOCUMENTATION BE A CRITERION FOR PRODUCT COMPLETION/ DELIVERY? (PAY NOW OR PAY LATER)

It is essential that the documentation be delivered as

part of the completed software product. Without the documentation, there is little assurance that the software satisfies stated requirements or that the organization will be able to maintain it. The cost of software maintenance is proportional to the effectiveness of the documentation which describes not only what the system does, but the logic used to accomplish its tasks.

30. WHY SHOULD AN ORGANIZATION HAVE A SOFTWARE MAINTENANCE STANDARDS POLICY?

A software maintenance standards policy helps to ensure software quality. Such standards describe in broad terms the responsibilities, authorities, functions, and operations of the software maintenance organization. The policy should be comprehensive enough to address any type of change to the software system and its environment, including changes to the hardware, software and firmware. To be effective, the policy should be consistently applied and must be supported and promulgated by upper management to the extent that it establishes an organizational commitment to software maintenance. When supported by management, the standards and guidelines help to direct attention toward the need for greater discipline in software design, development, and maintenance.

31. WHAT ARE SOME OF THE KEY ELEMENTS WHICH SHOULD BE INCLUDED IN A SOFTWARE PRODUCT RFP TO ADDRESS SOFTWARE MAINTENANCE?

Too often, Federal managers are forced to accept software products which, in addition to be delivered late, do not perform as required. An examination of the RFP usually reveals that:

- o performance clauses were omitted;
- o software quality assurance plans were not

required; and
- o software configuration management plans were left to the discretion of the contractor.

These are just a few of the elements that could help to prevent the delivery of defective, incomplete software products.

TECHNICAL

32. HOW CAN MAINTAINABILITY BE BUILT INTO EXISTING SOFTWARE?

There must be a program which has achieving maintainable software as its sole objective. It should include:

- o a plan for ensuring the maintainability requirements are specified into the software change design.
- o a measurement procedure to verify that maintainability goals have been met.
- o a performance review to provide feedback to managers, users, and maintainers on the effectiveness of the maintainability program.

33. WHAT ARE SOME STEPS THAT CAN BE TAKEN TO HELP IMPROVE SOFTWARE MAINTENANCE?

- o Use high level languages
- o Use standard coding conventions (variable names, structures, etc.)
- o Use modular structures
- o Use meaningful comments
- o Use only standard compiler options

34. WHAT CAN BE DONE TO TURN EXISTING SPAGHETTI CODE INTO UNDERSTANDABLE, RELIABLE SOFTWARE?

Software maintenance is a labor intensive activity. Consequently, automated tools should be used whenever possible. There are a number of structuring tools available that are designed specifically for reformatting and restructuring code. Some of these have been around for several years and thus, have a track record. For further details see [NBS88].

35. WHAT IS THE BEST MEASURE FOR DETERMINING IF THE QUALITY OF THE SOFTWARE IS DEGRADING?

If changes to specific areas begin to occur more frequently, and require increasingly more effort to correct or modify, the software is probably degrading. This is particularly true if someone familiar with the code is making the changes.

36. WHAT ARE METRICS?

A metric is a measure of the extent or degree to which a product possesses or exhibits a certain characteristic for determining the value or level of effort required to perform a given function.

37. HOW CAN METRICS BE USED TO IMPROVE THE SOFTWARE MAINTENANCE PROCESS AND PRODUCTS?

Currently, there are a number of metrics used to determine productivity. Of these, Lines of Code is the most commonly used The complexity metrics can also be useful for determining how difficult a software change will be. Other metrics that can be employed include automated completeness and consistency checkers. Not

only can they be used to measure programmer productivity, they can aid in understanding how software changes affect the overall software system from a maintainability standpoint

38. WHAT ARE THE KEY CONTRIBUTORS TO SOFTWARE MAINTENANCE COMPLEXITY?

- o Deeply nested DO loops
- o Excessive IF statements
- o Excessive use of global variables
- o Excessive GOTO statements
- o Embedded parameters, literals, constants
- o Self-modifying code
- o Excessive interaction between modules
- o Multiple entry-exit modules
- o Redundant modules

39. WHAT ARE THE "DRIVERS" OF SOFTWARE MAINTENANCE?

The primary driver of software maintenance is requests for enhancements. Other drivers include the poor condition of the code, scheduling, budget constraints, use of ineffective or outdated programming techniques, and tools

40. WHAT ARE SOME SOFTWARE MAINTENANCE TECHNIQUES WHICH HAVE PROVEN TO BE USEFUL?

- o Top down programming
- o Stepwise refinement
- o Regression testing
- o Code walkthroughs
- o Code audits/reviews
- o Peer reviews

41. IS DOCUMENTATION NECESSARY IN A MAINTENANCE ENVIRONMENT?

Documentation is valuable in both development and maintenance environments The level of documentation needed is a function of both the application and the maintenance environment For further information on documentation, see [FIPS38], [FIPS64], and [FIPS106].

42. AT WHAT POINT SHOULD DOCUMENTATION BE REQUIRED?

Documentation should be required for each product deliverable, intermediate as well as final

43. HOW SHOULD TEST DATA BE HANDLED?

There should be a well-defined policy for generating and storing test data In some instances, it is preferable to use actual data rather than generated data. In either case, the user should help develop the test data. For further information on testing,
see [NBS56], [NBS93], and [FIPS106]

44. WHEN SHOULD FORMAL LIBRARY PROCEDURES BE USED?

Microcomputers, especially when linked to mainframe computers, are making software systems increasingly accessible. One method of ensuring that the production or operational systems are not altered intentionally or unintentionally is to enforce formal library procedures. This usually can be accomplished by permitting adequate access to essential software and limited (read only) or no access to the operational software.

USERS

45. SHOULD USERS BE INVOLVED IN DEFINING OBJECTIVES OF THE SOFTWARE PRODUCT?

The days of casting off the users as uninformed are past. In todays' environment, there is a general recognition that user input is essential. Whether it is as a member of the configuration control board or the audit and review team, the key to producing a correct product is early user involvement

46. WHY SHOULD USERS BE INVOLVED IN THE CHANGE PROCESS?

Users have as much at stake as anyone in wanting the change to be implemented correctly. Therefore, their input should be solicited

47. SHOULD USERS, MANAGERS, AND MAINTAINERS BE INVOLVED IN DECISIONS REGARDING SOFTWARE CHANGES?

Yes. Too often the cause of software failure is due to inadequate communication between those who must use or are affected by the software change. A number of techniques can be employed to facilitate interface between users, maintainers, and managers. They include walkthroughs, configuration management control boards, audit and review teams, etc.

48. WHY SHOULD USERS BE INVOLVED IN GENERATING TEST DATA?

The users are more familiar with the data than the maintainer. While generated test data is useful, user generated, or live user data, is always preferable.

49. AS USERS ACQUIRE MORE AND MORE MICROCOMPUTERS AND ATTEMPT TO DO THEIR OWN PROGRAMMING, WILL THE SOFTWARE MAINTENANCE BURDEN INCREASE?

Users do not always know in detail what they want until they see a version of it, and then they want to modify. Unless there is close interaction between users and data processing personnel the gap between these two groups will widen as users seek to handle more of their application development and maintenance. In the short term, the burden will be somewhat lessened, but as users experiment with more sophisticated packages the maintenance burden on the traditional data processing departments is likely to increase.

50. WHAT ARE SOME SUCCESSFUL SOFTWARE MAINTENANCE TOOLS?

Tools that have been found to be effective include: automated restructurers, debuggers, documenters, test and program generators, editors, cross referencers software configuration management, and tracing tools.

SOFTWARE CONFIGURATION MANAGEMENT (SCM)

51. WHAT IS SOFTWARE CONFIGURATION MANAGEMENT?

Software Configuration Management (SCM) refers to the control of software changes. It helps to ensure that:

- o all software change requests are handled accurately and completely;
- o the resulting products satisfy the specified requirements;
- o key software maintenance considerations,

responsibilities, and requirements are identified; and

- o the processing of software change requests (SCR) is facilitated.

52. WHAT ARE THE SCM RESPONSIBILITIES?

SCM is responsible for configuration identificatio configuration control, configuration status accountin auditing, records retention, disaster recovery, libra activities, and coordinating the various activities wi the users, management, and the staff.

53. DOES SCM APPLY TO THE SOURC CODE AND DOCUMENTATION?

SCM should be applied to the baseline documentatio as well as the source and object code.

54. WHAT TOOLS SUPPORT SCM?

There are a number of tools that are effective for SCM Most of these tools provide version control and/or l brary and archival functions. However, there is a wi range in the price of this class of tools. For mo specific information, see [NBS88].

55. ARE TOOLS ESSENTIAL FOR IMPLE MENTING SCM?

Manually tracking software changes back to the r quirements and to other changes is a very labo intensive process. In a large application environmen with frequent changes, performing SCM manually ca prove to be quite difficult, if not impossible.

53. WHAT IS A CONFIGURATION CONTROL BOARD (CCB)?

The primary role of the configuration control board (CCB) is to facilitate software changes into the system. The CCB should be comprised of representatives who are involved in, or affected by, the software changes.

57. WHAT ARE THE SPECIFIC DUTIES OF THE CCB?

- o Evaluate, assign, prioritize, and schedule software maintenance work requests.
- o Set the meeting time.
- o Establish the agenda to consider proposed software changes plans, procedures, and interfaces.
- o Inform the initiator of the software change request on the actions taken.
- o Track progress of all maintenance tasks and ensure that they are on or ahead of schedule.
- o Adjust schedules when necessary.
- o Communicate progress and problems to the user.
- o Communicate progress and problems to upper management.
- o Establish and maintain maintenance standards and guidelines.
- o Enforce standards and make sure that the software maintenance is of high quality.

58. WHAT LEVEL OF FORMALITY IS NEEDED TO IMPLEMENT AN SCM PROGRAM?

The level of formality depends on the environment. There may be a CCB or an individual who functions as a CCB. The key is to assign the responsibility for ensuring that the software changes satisfy the stated requirements before they are released for production.

TOOLS

59. WHAT STEPS SHOULD BE TAKEN PRIOR TO SELECTING TOOLS FOR SOFTWARE MAINTENANCE?

- o There should be an inventory of the existing tools.
- o Current and potential use of the tools should be examined
- o An attempt should be made to acquire tools that are compatible with the existing environment.
- o Training should be planned for everyone who will either use tools or be affected by thier use.

60. WHAT SHOULD BE CONSIDERED WHEN SELECTING TOOLS?

Many tools have been developed in a research environment and used as prototypes to demonstrate concepts. They were never engineered as production products though they are being used as such. Such tools are generally not well documented, not efficiently coded, seldom portable, and often not adequately supported.

61. WHAT IS AN INTEGRATED TOOL?

An integrated tool generally refers to one that uses a common data base and/or a common command language for controlling and using a set of tools. The use of such tools may help to enforce uniformity within a software maintenance environment.

62. HOW ARE SOFTWARE MAINTENANC TOOLS CATEGORIZED?

Software maintenance tools fall into seven categories:

- o Configuration management
- o Monitoring/evaluation
- o Redesign
- o Code production/analysis
- o Verification/validation/testing (VVT)
- o Testing/integration
- o Documentation

63. WHAT ARE SOME EXAMPLES O SOFTWARE MAINTENANCE TOOLS I EACH CATEGORY?

Configuration management:
support library, status reporting
Monitoring/evaluation:
performance analyzers, automatic recovery tools
Redesign:
requirements analyzers
Code production/analysis:
structurer, debugger, comparator
VV&T:
static an is, path analyzer
Testing:
test data generator
Documentation:
automatic documentor

64. CAN SOFTWARE MAINTENANCE BE CONTROLLED?

In order to maintain control over the software maintenance process, and to ensure that the maintainability of the system does not deteriorate, it is important that software maintenance be anticipated and planned for. The quality and maintainability of a software system often decrease as the system grows older. This is the result of many factors which, taken one at a time, may not seem significant but become cumulative and often result in a system which is very difficult to maintain. Quality programming capabilities and techniques are readily available. However, until a firm discipline is placed on how software maintenance is performed, and that discipline is enforced, many systems will be permitted to deteriorate to the point where they are impossible to maintain.

Software maintenance must be performed in a structured, controlled manner. It is not enough to get a system "up and running" after it breaks. Proper management control must be exercised over the entire process. In addition to controlling the budget, schedule, and staff, it is essential that the software maintenance manager control the system and the changes to it. Systems must not only be developed with maintenance in mind, they must be maintained with maintainability in mind. If this is done, the quality and maintainability of the code actually can improve.

SUPPORTING ICST DOCUMENTS

[FIPS38] "Guidelines for Documentation of Computer Programs and Automated Data Systems," FIPS PUB 38, 1976.

[FIPS64] "Guidelines for Documentation of Computer Programs and Automated Data Systems for the Initiation Phase," FIPS PUB 64, 1979.

[FIPS99] "Guideline: A Framework for the Comparison of Software Development Tools," FIPS Pub 99, 1983.

[FIPS101] "Guideline for Lifecycle Validation, Verification, and Testing of Computer Software," FIPS PUB 101, 1983.

[FIPS105] "Guideline for Software Documentation Management," FIPS PUB 105, 1984.

[FIPS106] "Guideline on Software Maintenance," FIPS PUB 106, 1984.

[NBS56] NBS Special Pub 500-56 "Validation, Verification, and Testing for the Individual Programmer," M.Branstad, J.Cherniavsky, and W.Adrion, 1980.

[NBS78] NBS Special Pub 500-78,"NBS Programming Environment Workshop Report," M.Branstad and W.R.Adrion, eds.,1978.

[NBS87] NBS Special Pub 500-87 "Management Guide to Software Documentation," A.Neumann, 1982.

[NBS88] NBS Special Pub 500-88 "Software Development Tools," R.Houghton, Jr., 1982.

SUPPORTING NBST DOCUMENTS

[NBS93] NBS Special Pub 500-93 "Software Validation, Verification, and Testing Technique and Tool Reference Guide," P.Powell,Editor, 1982.

[NBS106] NBS Special Pub 500-106 "Guidance on Software Maintenance," R.Martin and W.Osborne, 1983.

[NBS114] NBS Special Pub 500-114 "Introduction to Software Packages," Sheila Frankel, editor, 1984.

SUGGESTED ADDITIONAL READING

[BOEH81] B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.

[COUG82] D.J.Couger and M.A.Colter, "Effect of Task Assignments on Motivation of Programmers and Analysts," research report, Univ. of Colorado, 1982.

[GLAS81] R.L.Glass and R.A.Noiseux, *Software Maintenance Guidebook*, Prentice Hall,1981.

[MART83] J Martin, C.McClure, *Software Maintenance - The Problem and Its Solutions*, Prentice Hall, 1983.

[MCCL81] C.L McClure, *Managing Software Development and Maintenance*, Van Nostrand Reinhold, 1981.

[PARI80] G.Parikh, editor, *Techniques of Program and System Maintenance*, Ethnotech, 1980.

[PARI83] G.Parikh, N.Zvegintzov, *Tutorial on Software Maintenance*, IEEE Computer Society Press, 1983.

[PERR81] W.E.Perry, *Managing System Maintenance*, Q.E.D. Information Sciences, Inc., 1981.

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series. National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)